A Collective, Probabilistic Approach to Schema Mapping Using Diverse Noisy Evidence

Angelika Kimmig, Alex Memory, *Member, IEEE,* Renée J. Miller, *Member, IEEE,* and Lise Getoor, *Member, IEEE*

Abstract—We propose a probabilistic approach to the problem of schema mapping. Our approach is declarative, scalable, and extensible. It builds upon recent results in both schema mapping and probabilistic reasoning and contributes novel techniques in both fields. We introduce the problem of *schema mapping selection*, that is, choosing the best mapping from a space of potential mappings, given both metadata constraints and a data example. As selection has to reason holistically about the inputs and the dependencies between the chosen mappings, we define a new schema mapping optimization problem which captures interactions between mappings as well as inconsistencies and incompleteness in the input. We then introduce *Collective Mapping Discovery* (**CMD**), our solution to this problem using state-of-the-art probabilistic reasoning techniques. Our evaluation on a wide range of integration scenarios, including several real-world domains, demonstrates that **CMD** effectively combines data and metadata information to infer highly accurate mappings even with significant levels of noise.

1 Introduction

Schema mappings are collections of complex logical statements which relate multiple relations across data sources with different schemas, and thus can be used to exchange data between these sources. Efficient techniques for reasoning about the suitability of different schema mappings are crucial to manage the massive number, complexity, and size of data sources. While the metadata and data of the sources often provide evidence for how to best map them, this evidence is rarely complete or unambiguous. To reason effectively about mappings, we thus need techniques grounded in mapping understanding that can reason about open-world scenarios using uncertain, imperfect evidence.

We study the problem of *mapping selection*, that is, of selecting from a large set of possible mappings, a mapping that best relates a source and a target schema. We define the mapping selection problem for the entire language of *st tgds* (source-to-target tuple-generating-dependencies; also known as GLAV mappings) which is arguably the most commonly used mapping language [1]. We prove that exactly solving this problem is NP-hard already for full st tgds, i.e., st tgds without existential quantifiers. We then provide an efficient and highly accurate approximate solution to this problem based on state-of-the-art probabilistic reasoning.

Historically, approaches to schema mapping discovery and selection have considered a wide variety of inputs. Early approaches use metadata (schema constraints) and attribute correspondences (aka schema matchings) to create mappings that are consistent with the metadata [2], [3]. Metadata in the form of query logs has been used to select mappings that are most consistent with frequently asked queries [4]. Many different approaches use data to refine a mapping or to select a mapping from among a set of schema mappings [5], [6], [7], [8], [9], [10], [11], [12]. Other approaches solicit user feedback to define scores for each view in a set of candidate views and then select an optimal set of views based on these scores [13]. All of these approaches have merit, but are tailored to a specific form of input evidence, and either work for limited mapping languages, like views, or assume consistent or complete input, which is difficult to prepare or find. An exception to this is the approach by Alexe et al. [12] that considers *bad* data examples that are consistent with several (candidate) mappings or none. They consider how such *bad* examples can be turned into *good* examples that are consistent with a single, desired mapping.

We define a new mapping selection problem that uses both data and metadata collectively as input. None of the evidence is required to be consistent or complete, rather we find the subset of st tgds that are best supported by the given evidence as a whole. Metadata can serve as a guide through a potentially massive set of possible mappings, suggesting mappings that are consistent with schema semantics (e.g., joining relations on a foreign key). Data can reinforce metadata evidence. Data can also rule out a mapping that is consistent with the metadata, but inconsistent with large parts of the data. Metadata can obviate the need to have two pristine data instances as input that precisely define a single *best* mapping. Furthermore, our framework is declarative and extensible to new forms of evidence including scores (such as user-feedback annotations) on the metadata and data evidence.

Our solution adopts and extends some of the latest techniques from the probabilistic reasoning community. These techniques are routinely used to combine logical constraints in relational domains with the ability to handle uncertainty

- *A. Kimmig is with the School of Computer Science and Informatics, Cardiff University. E-mail: KimmigA@cardiff.ac.uk*
- *A. Memory is with Department of Computer Science, University of Maryland. E-mail: memory@cs.umd.edu*
- *R. Miller is with the College of Computer and Information Science, Northeastern University. E-mail: miller@northeastern.edu*
- *L. Getoor is with University of California Santa Cruz. E-mail: getoor@ucsc.edu*

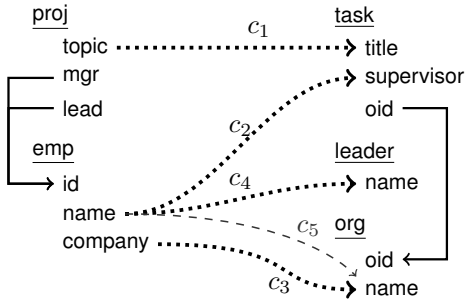
Manuscript received April XX, XXXX; revised August XX, XXXX.

and conflicting information. Building upon work of Gottlob and Senellart [18], we refine their concepts of validity and fully explaining to define what it means for a single tuple to be either an (incomplete) error for a mapping or (partially) explained by a mapping. Using these notions, we define our probabilistic optimization problem using probabilistic soft logic (PSL) [14], a scalable probabilistic programming language based on weighted logical rules. PSL has been used successfully for a variety of data and knowledge integration problems, including knowledge graph identification [15] and data fusion [16], [17]. It however did not support the kind of open world reasoning required for mapping selection, where we need to express constraints over the existence of elements in a set satisfying certain conditions, namely, st tgds in the mapping explaining tuples in the data example, and furthermore, preferences over these elements are available. We therefore extend PSL with *prioritized disjunctions*, which provide a tractable framework for handling such existential, weighted constraints, and thereby allow us to define key features of the mapping selection problem. To use data and metadata as input, we use the extended PSL language as a common representation for both. The data evidence comprises a data example and the metadata evidence comprises a set of st tgds. By having a common language for reasoning, we can easily integrate data and metadata evidence by, for example, reasoning about whether a data example satisfies metadata evidence such as part of a mapping.

We refer to our solution as *Collective Mapping Discovery* (**CMD**), because it reasons collectively both about multiple forms of evidence and over the interactions between different st tgds. **CMD** advances the state-of-the-art in schema mapping by using more kinds of evidence and integrating them at a much finer-grained level of detail than attempted in the past. In addition, the declarative nature of **CMD** makes it easy to extend in a variety of ways.

We perform an extensive empirical validation of our approach. We use the integration benchmark iBench [18] to test **CMD** on a wide variety and large number of mapping scenarios. We use IQ-METER [19], a multi-criterion evaluation measure, to confirm the quality of **CMD**'s output. We compare **CMD** with a baseline approach which uses only metadata. We show that the accuracy of **CMD** is more than 33% above that of a metadata-only approach already for small data examples. We illustrate the robustness of our approach by demonstrating that we are able to find accurate mappings even if a quarter of the data is dirty. We demonstrate that the approach scales well with the size of both metadata and data, and effectively selects small, correct mappings even if dozens of competing candidate mappings are available for each tuple. In addition, we show that **CMD** is effective on several problems with real data.

This paper expands on our earlier work [20], proving complexity results, and contributing experimental results on additional real world problems as well as problems with an order of magnitude greater complexity. Section 2 illustrates the key challenges with an example. Section 3 introduces the selection problem for st tgds without existentially quantified variables, and Section 4 extends this to st tgds. Section 5 introduces our solution using PSL and our extension of PSL with *prioritized disjunctions*. We discuss experiments in Section 6 and related work in Section 7.



(a) Source (left) and target schema (right) with corresponding attributes (dotted lines), a spurious correspondence (dashed), and foreign key constraints (solid lines).

proj

topic	mgr	lead
BigData	1	2
ML	1	1

task

title	supervisor	oid
BigData	Alice	111
ML	Alice	111

emp

id	name	company
1	Alice	SAP
2	Bob	IBM
3	Pat	MS

(b) Initial data example.

leader

name
Alice
Bob

org

oid	name
111	SAP
222	MS

(c) Additional data.

θ_0 : proj$(\mathbf{t}, \mathrm{m}, \mathrm{l}) \wedge$ emp$(\mathrm{m}, \mathbf{n}, \mathrm{c}) \rightarrow \exists\ \mathrm{o}.$ task$(\mathbf{t}, \mathbf{n}, \mathrm{o})$

θ_1 : proj$(\mathbf{t}, \mathrm{m}, \mathrm{l}) \wedge$ emp$(\mathrm{l}, \mathbf{n}, \mathrm{c}) \rightarrow \exists\ \mathrm{o}.$ task$(\mathbf{t}, \mathbf{n}, \mathrm{o})$

θ_2 : proj$(\mathbf{t}, \mathrm{m}, \mathrm{l}) \wedge$ emp$(\mathrm{m}, \mathbf{n}, \mathbf{c}) \rightarrow \exists\ \mathrm{o}.$ task$(\mathbf{t}, \mathbf{n}, \mathrm{o}) \wedge$ org$(\mathrm{o}, \mathbf{c})$

θ_3 : proj$(\mathbf{t}, \mathrm{m}, \mathrm{l}) \wedge$ emp$(\mathrm{l}, \mathbf{n}, \mathbf{c}) \rightarrow \exists\ \mathrm{o}.$ task$(\mathbf{t}, \mathbf{n}, \mathrm{o}) \wedge$ org$(\mathrm{o}, \mathbf{c})$

θ_4 : emp$(\mathrm{i}, \mathrm{n}, \mathbf{c}) \rightarrow \exists\ \mathrm{o}.$ org$(\mathrm{o}, \mathbf{c})$

θ_5 : emp$(\mathrm{i}, \mathbf{n}, \mathrm{c}) \rightarrow$ leader$(\mathbf{n})$

θ_6 : proj$(\mathrm{t}, \mathrm{m}, \mathrm{l}) \wedge$ emp$(\mathrm{l}, \mathbf{n}, \mathrm{c}) \rightarrow$ leader$(\mathbf{n})$

θ_7 : proj$(\mathbf{t}, \mathrm{m}, \mathrm{l}) \wedge$ emp$(\mathrm{m}, \mathbf{n}, \mathrm{c}) \rightarrow \exists\ \mathrm{o}.$ task$(\mathbf{t}, \mathbf{n}, \mathrm{o}) \wedge$ org$(\mathrm{o}, \mathbf{n})$

θ_8 : proj$(\mathbf{t}, \mathrm{m}, \mathrm{l}) \wedge$ emp$(\mathrm{l}, \mathbf{n}, \mathrm{c}) \rightarrow \exists\ \mathrm{o}.$ task$(\mathbf{t}, \mathbf{n}, \mathrm{o}) \wedge$ org$(\mathrm{o}, \mathbf{n})$

(d) Candidate st tgds. Variables in bold denote exchanged attributes.

Fig. 1: Motivating example; see Section 2 for details.

2 MOTIVATING EXAMPLE

Figure 1(a) shows a pair of source and target schemas, foreign keys (solid lines) and attribute correspondences (or matches, dotted lines), which we will use as a running example. The metadata is ambiguous, as it is not clear from the schemas whether task.supervisor in the target schema is associated with proj.mgr or proj.lead in the source schema. A data example in the form of an instance of the source schema

(I) and an instance of the target schema (J) can help resolve such ambiguity. The data example in Figure 1(b), where org and leader are empty, suggests that supervisors in task tuples correspond to mgr in the source, not lead. Interactive schema mapping refinement techniques use data to select among a set of mappings. They take as input a set of candidate mappings and use data to interactively guide a user in selecting a subset that is correct [5], [21], or in correcting a set of data examples so that a "best fitting" mapping exists [22]. The interactive nature of these solutions permits a user to decide what mapping is best given metadata and data evidence. In contrast, we do this reasoning automatically to find the best fitting mapping.

We consider the problem of combining metadata evidence (in the form of a set of candidate mappings) and potentially imperfect data evidence (in the form of a data example) to select an optimal mapping. More specifically, our candidate mappings are source-to-target tuple-generating-dependencies (st tgds).[1] These are simple first-order logic statements relating a source query and a target query. The candidates may come from a mapping design tool like Clio [23] or ++Spicy [24], or may have been mined from a query log [4].

A key challenge in mapping selection is that the number of possible selections is exponential in the number of candidate st tgds. Consider the candidates in Figure 1(d), focusing first on our earlier data example (Figure 1(b)) and candidates θ_0 and θ_1. Notice that the data example is *valid* for θ_0 (meaning (I,J) satisfy the mapping θ_0) but is not *valid* with respect to θ_1, as there is no tuple (BigData, Bob, -) (with some oid). We call such a missing tuple an *error*. Errors might be caused by dirty data. The data example contains a tuple (BigData, Alice, 111) and this tuple may be dirty (the value Alice is wrong and should be Bob) causing this *error*. If the data is clean, this error tuple would suggest that we should prefer θ_0 over θ_1.

Note that θ_0 and θ_1 both ignore the correspondence between emp.company and org.name. Mapping θ_2 also explains the data (intuitively), but it explains more, as it creates org tuples for which we have no data evidence. If we change our data example to include the org tuples in Figure 1(c), the data suggests that we should select both θ_2 and θ_4. The mapping θ_2 alone maps the inner join of the source data to the target. Mappings θ_2 and θ_4 together map the right outer-join.

If we also add the leader tuples in Figure 1(c) to our data example, θ_5 explains all leader tuples. However, θ_5 is not *valid* with respect to the data, as it also suggests that tuple (Pat) should appear in leader, but it does not and thus is an error for θ_5. The mapping θ_6 addresses this by joining emp with proj via proj.lead; it both explains and is valid with respect to the leader example data. Generally, we seek sets of st tgds that collectively explain the data and are valid with respect to the data. On that basis, the set $\{\theta_2, \theta_4, \theta_6\}$ is a good choice.

Note that our candidates θ_0 - θ_6 use only correspondences c_1-c_4 in Figure 1(a). If a matcher incorrectly suggested correspondence c_5, then we may get additional candidate mappings like θ_7 or θ_8 that use this correspondence. However, in this example (and in many real examples) a small data example can eliminate such candidates, as they are likely not to explain the data or be valid.

This example illustrates many challenges in schema mapping discovery from metadata and data.

DIRTY OR AMBIGUOUS METADATA. Our goal is to find a mapping that fits the metadata. In practice, the number of such mappings can be huge, due to metadata ambiguities such as 1) multiple foreign key paths between relations; 2) the choice between inner and outer joins; 3) the presence of bad correspondences. Dirty metadata (for example, incorrect foreign keys) exacerbates this problem. Data can help in selecting correct mappings. We tackle the problem of combining metadata and data evidence to effectively and efficiently select a mapping, even if the data does not fully disambiguate all metadata. In our example, we may have some target tuples that are consistent with a join on mgr (θ_0) and some that are consistent with a join on lead (θ_1); e.g., (ML, Alice, 111) is consistent with both θ_0 and θ_1. Our solution will weigh the evidence to find a mapping that is most consistent with the evidence as a whole.

UNEXPLAINED DATA. We are given example source (I) and target (J) data and our goal is to find a mapping that explains the target data. In practice, we rarely have *perfect* data examples that only contain target data explained by I. Indeed, the open nature of st tgds permits the target to have independent data that was not derived from the source. For example, suppose there is a target org (333, BM), and the value BM does not appear in the source. This data may be correct data (the target has data about the Bank of Montreal and the source does not) or it may be dirty data (perhaps the value BM was mistyped and should be IBM). Even if no candidate explains these tuples, we still want to find the best mapping. So our optimization should not fail in the presence of such *unexplained tuples*. Furthermore, if there is a mapping that explains all data, we may not choose it if it is not valid with respect to the data example, or if it is considerably more complex than one that fails to explain a few tuples in J.

DATA ERRORS. Our goal is to find a mapping that is valid for the given data example (I, J). Again, in practice, it is unrealistic to assume a data example that is *perfect* in this way. Hence, we provide a solution that is tolerant of some errors (for example, some dirty source data or some missing target data), but seeks to find a set of st tgds for which the errors are minimized.

UNKNOWN VALUES. Our goal is to find a mapping that may use existential quantification where appropriate. This is challenging, as such mappings introduce unknown or null values in the target. For instance, st tgds θ_0 and θ_1 both only cover part of target tuple (ML, Alice, 111), as they cannot "guess" the value of the oid. Still, we need to compare them to st tgds that may have no existentials and therefore cover entire target tuples. This problem is made more challenging as existentials play a critical role in identifier (or value) invention where the same existential value is used in multiple tuples to connect data together. It is important that mappings that correctly connect the data be considered better than mappings that use different existentials. For example, we prefer θ_2 over the combination of θ_0 and θ_4,

1. The term *mapping* is often used both for a single st tgd and for a set of st tgds. Here, we use *candidate mapping* or *candidate* to refer to a single st tgd; while *mapping* generally refers to a set of st tgds.

since the data supports the connection θ_2 makes between task and org in the target. This is an important aspect of the problem that has not been considered by earlier work on view (also known as full st tgd) selection [13].

To address these challenges, we present a fine-grained, scalable solution that gives an st tgd *credit* for each tuple it can explain or partially explain (in the case of existential mappings) and aggregates this information to find a set of st tgds that best explain the data. A set of st tgds is penalized for each error tuple (the more errors the less valid the mapping). Hence, we find the set of candidate st tgds that collectively minimize the number of errors and number of unexplained tuples, even under contradictory or incomplete evidence.

3 Selection over Full Mappings

We first define mapping selection for full st tgds [1], that is, st tgds without existentially quantified variables, and extend our definitions to arbitrary st tgds in Section 4.

3.1 Mapping Selection Inputs

We define our mapping selection problem with respect to a *source* schema **S** and a *target* schema **T**, where a schema is a set of relations. The *data evidence* consists of a data example, that is, a pair of instances I of **S** and J of **T**. The *metadata evidence* consists of a (finite) set $\mathcal{C}$ of candidate st tgds. An st tgd is a logical formula $\forall \mathbf{x}\ \phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y})$, where ϕ is a conjunction of atoms over the relations of **S** and ψ over those of **T** [1]. Here, **x** and **y** are sets of logical variables. If **y** is empty (no existentials) then the st tgd is a *full st tgd* [25].

Candidate st tgds can be generated using existing schema mapping systems. Such systems, both industrial systems and research systems, generate sets of candidate mappings and generally let users select or refine these mappings using a variety of visual interfaces. To generate candidate mappings, research systems like Clio [23], HepTox [26], and ++Spicy [24] use schema constraints, while U-Map [4] uses query logs. By building on these existing approaches, we focus on candidate mappings that are plausible according to the metadata and the methodology used in candidate generation rather than all possible mappings.

3.2 Characterizing the Input Quality

Given metadata evidence (**S**, **T**, $\mathcal{C}$), our goal is to find a subset $\mathcal{M} \subseteq \mathcal{C}$ that best "fits" the data example (I, J). Let **C** be the set of all constants in $I \cup J$, and **N** a set of labeled nulls (disjoint from **C**). Following Fagin et al. [25], a homomorphism between instances $h : K_1 \rightarrow K_2$ is a mapping from $\mathbf{C} \cup \mathbf{N}$ to $\mathbf{C} \cup \mathbf{N}$ such that: (1) for every $c \in \mathbf{C}$, $h(c) = c$, and (2) for every $R(t)$ of K_1, $R(h(t))$ is in K_2. A homomorphism $h : \phi(x) \rightarrow K$ is a mapping from the variables x to $\mathbf{C} \cup \mathbf{N}$ such that for every $R(x_1, \ldots, x_n)$ in $\phi(x)$, $R(h(x_1), \ldots, h(x_n))$ is in K. Let $\mathcal{M}$ be a set of st tgds, then an instance K of **T** is called a *solution* for I, if $(I, K) \models \mathcal{M}$. An instance K is a *universal solution* if it is a solution and if for every other solution K', there is a homomorphism $h : K \rightarrow K'$. Fagin et al. [25] showed how a universal solution can be computed efficiently using the chase over $\mathcal{M}$ (and such a universal solution is typically called a *canonical universal solution*).

Gottlob and Senellart [7] call a mapping $\mathcal{M}$ *valid* for (I, J) if J is a solution for I under $\mathcal{M}$. Suppose $(I, J) \not\models \mathcal{M}$. Intuitively, this means J misses tuples that must be in every solution for I. We call such tuples *errors*. A ground tuple t (that is, a tuple containing only constants) is a *full error* if it is not in J but in every J' such that $(I, J \cup J') \models \mathcal{M}$. If K is a universal solution for $\mathcal{M}$ and I, then t is a full error iff $t \in K$ and $t \notin J$. If (I, J) is valid with respect to $\mathcal{M}$ then there are no full errors.

Example 1: *The candidate θ_5 in Figure 1(d) is not valid with respect to the data example in Figure 1(c). However, if we add the tuple $t' =$ leader(Pat) to J then θ_5 is valid for $(I, J \cup t')$. Thus, (Pat) is a* full error, *and the only full error.* □

Ideally, all tuples in J should be explained, that is, be a result of the selected candidate mappings applied to I. Again following Gottlob and Senellart [7], a mapping $\mathcal{M} \subseteq \mathcal{C}$ and source instance I *explain* a ground fact t, if $t \in K$ for every K such that $(I, K) \models \mathcal{M}$. A mapping $\mathcal{M}$ and I *fully explain* J if they explain every tuple in J. A ground tuple t is *explained* by $\mathcal{M}$ and I iff t is in a universal solution for $\mathcal{M}$ and I, else t is an *unexplained tuple*. As with validity, we would like to permit exceptions, that is, a few tuples in J that are unexplained, meaning J is not fully explained.

Example 2: *Consider again θ_5 of Figure 1(d). For the instance I of emp shown in (b), θ_5 fully explains J (the two leader tuples) shown in (c). However, if leader also contained leader(Joe), then θ_5 would still be valid, but leader(Joe) is an unexplained tuple.* □

3.3 Collective Selection over Full Mappings

We now define an optimization problem for finding a mapping $\mathcal{M} \subseteq \mathcal{C}$ that best fits our imperfect evidence by jointly minimizing:

1) the number of unexplained tuples;
2) the number of error tuples; and
3) the size of $\mathcal{M}$.

The first two are formalized through functions that, for a candidate set $\mathcal{M}$, compare the given target instance J to the solution for $\mathcal{M}$ and I, i.e., check how many tuples in J are unexplained (collectively) by $\mathcal{M}$, and how many tuples resulting from data exchange with each st tgd in $\mathcal{M}$ are not in J. Figure 2 illustrates these different kinds of tuples.

Let $K_{\mathcal{C}}$ (respectively, $K_{\mathcal{M}}$ and K_θ) be a canonical universal solution for I and $\mathcal{C}$ (respectively, $\mathcal{M}$ and θ). We consider full st tgds so canonical universal solutions are unique. We define $\mathsf{creates}_{\mathsf{full}}(\theta, t)$ as follows:

$$\mathsf{creates}_{\mathsf{full}}(\theta, t) = \begin{cases} 1 & t \in K_\theta \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

We then define $\mathsf{error}_{\mathsf{full}}(\mathcal{M}, t)$ for a tuple $t \in K_{\mathcal{C}} - J$ (Figure 2 left side) to be the number of st tgds in $\mathcal{M}$ for which t is an error.

$$\mathsf{error}_{\mathsf{full}}(\mathcal{M}, t) = \sum_{\theta \in \mathcal{M}} (\mathsf{creates}_{\mathsf{full}}(\theta, t)) \tag{2}$$

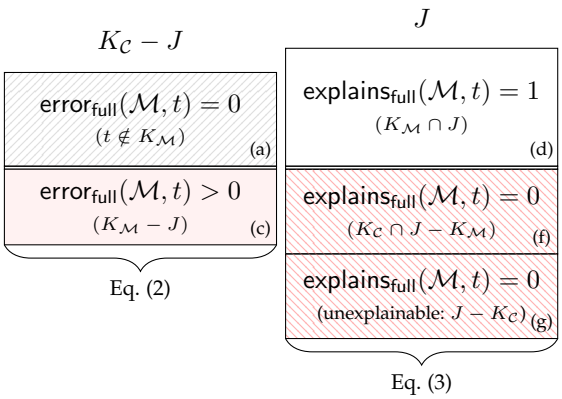


Fig. 2: Illustration of functions $\mathsf{error}_{\mathsf{full}}(\cdot)$ and $\mathsf{explains}_{\mathsf{full}}(\cdot)$.

Correspondingly, for the tuples in J (Figure 2 right side), we define the function $\mathsf{explains}_{\mathsf{full}}(\mathcal{M}, t)$, which checks whether such a tuple is explained by $\mathcal{M}$.

$$\mathsf{explains}_{\mathsf{full}}(\mathcal{M}, t) = 1 \text{ if } t \in J \cap K_{\mathcal{M}} \text{ and } 0 \text{ otherwise} \quad (3)$$

We call tuples in $J - K_C$ that cannot be explained by any st tgd in $\mathcal{C}$ *unexplainable tuples* (Figure 2(g)).

Finally, we define the size function $\mathsf{size}(\mathcal{M})$ to be the sum of the number of atoms in each $\theta \in \mathcal{M}$.

$$\mathsf{size}(\mathcal{M}) = \sum_{\theta \in \mathcal{M}} (\text{number atoms in } \theta) \quad (4)$$

This choice of complexity term provides a guard against including st tgds with insufficient support from the evidence. Taking these three criteria together, we formally define the *mapping selection problem for full st tgds* as follows.

Given schemas $\mathbf{S}$, $\mathbf{T}$, a data example (I, J), and a set $\mathcal{C}$ of candidate *full* st tgds

$$\textbf{Find} \quad \operatorname*{argmin}_{\mathcal{M} \subseteq \mathcal{C}} \Big(\sum_{t \in J} [1 - \mathsf{explains}_{\mathsf{full}}(\mathcal{M}, t)] + \sum_{t \in K_C - J} [\mathsf{error}_{\mathsf{full}}(\mathcal{M}, t)] + \mathsf{size}(\mathcal{M}) \Big) \quad (5)$$

As we show in Section 3.4, this problem is NP-hard.

Notice the similarity of the *mapping selection problem* with the formal framework for schema mapping discovery of Gottlob and Senellart [7]. They propose a way of *repairing* a mapping to (1) explain unexplained tuples and to (2) make the mapping valid for an invalid data example (in other words, to account for error tuples). They define an optimal mapping as one that minimizes a cost function containing three parts: the size of the mapping; the number of the repairs needed to account for unexplained tuples; and the number of repairs needed to account for error tuples. In contrast, we are counting error and unexplained tuples rather than using algebraic operators to repair the mapping. We weight each of these three components equally in our problem definition. However, our formalization permits each part to be weighted differently if there is *a priori* knowledge of the scenarios.

In terms of Figure 2, our goal is to find an $\mathcal{M}$ that jointly minimizes the number of unexplained but explainable tuples (those in (f)), the number of errors (those in (c)) and the size of $\mathcal{M}$. Note that every $\mathcal{M} \subseteq \mathcal{C}$ receives a constant penalty for unexplainable tuples (the tuples in $J - K_C$ (g)). These tuples can easily be removed for efficiency before running the optimization.

Note a subtle but important difference in how we treat errors and unexplained tuples. The definition of $\mathsf{error}_{\mathsf{full}}(\cdot)$ considers each candidate in $\mathcal{M}$ individually, and sums the number of errors made by each. That is, if two st tgds $\theta_i \in \mathcal{M}$ and $\theta_j \in \mathcal{M}$ both make an error on t, that error is counted twice. In other words, we seek a mapping where as few as possible of the st tgds in the mapping make an error on t. In contrast, we do not require each st tgd in the mapping to explain all tuples in J, but consider it sufficient if at least one $\theta \in \mathcal{M}$ explains a tuple. Thus, we cannot treat each θ individually, but we must reason about the set $\mathcal{M}$ as a whole.

3.4 Mapping Selection is NP-hard

The $\mathsf{error}_{\mathsf{full}}(\cdot)$ and $\mathsf{size}(\cdot)$ terms of (5) are modular and act as constraints on the supermodular $\mathsf{explains}_{\mathsf{full}}(\cdot)$ term. Such minimization tasks are often NP-hard, and we provide proof that this is also the case for our selection problem.

Theorem 1. *The mapping selection problem for full st tgds as defined in* (5) *is NP-hard.*

Proof. We use a reduction from SET COVER, which is well known to be NP-complete, and is defined as follows:
Given a finite set U, a finite collection $R = \{R_i \mid R_i \subseteq U, 1 \leq i \leq k\}$ and a natural number $n \leq k$, is there a set $R' \subseteq R$ consisting of at most n sets R_i such that $\bigcup_{R_i \in R'} R_i = U$?

We first consider the decision variant of mapping selection, which is defined as follows:
Given schemas $\mathbf{S}$, $\mathbf{T}$, a data example (I, J), a set $\mathcal{C}$ of candidate *full* st tgds, and a natural number m, is there a selection $\mathcal{M} \subseteq \mathcal{C}$ with $F(\mathcal{M}) \leq m$?
where $F(\mathcal{M})$ is the function minimized in (5), i.e.,

$$F(\mathcal{M}) = \sum_{t \in J} [1 - \mathsf{explains}_{\mathsf{full}}(\mathcal{M}, t)] + \sum_{t \in K_C - J} [\mathsf{error}_{\mathsf{full}}(\mathcal{M}, t)] + \mathsf{size}(\mathcal{M}) \quad (6)$$

We construct a mapping selection decision instance from a SET COVER instance as follows. We set $m = 2n$, introduce an auxiliary domain $D = \{1, \ldots, m+1\}$, and define

$$\begin{aligned} \mathbf{S} &= \{R_i/2 \mid R_i \in R\} \\ \mathbf{T} &= \{U/2\} \\ \mathcal{C} &= \{R_i(X, Y) \rightarrow U(X, Y) \mid R_i \in R\} \\ J &= \{U(x, y) \mid (x, y) \in U \times D\} \\ I &= \bigcup_{R_i \in R} \{R_i(x, y) \mid (x, y) \in R_i \times D\} \end{aligned}$$

We use notation R/k to indicate relation R has arity k. It is easily verified that this construction is polynomial in the size of the SET COVER instance. It is easily verified that this construction is polynomial in the size of the SET COVER instance. We next show that the answers to SET COVER and the constructed mapping selection problem coincide.

For each R_i, the candidate st tgd $\theta_i = R_i(X,Y) \rightarrow U(X,Y)$ has size two, makes no errors (as $R_i \subseteq U$), and for each $x \in R_i$ explains the tuples $U(x,1),\ldots,U(x,m+1)$. We thus have

$$F(\mathcal{M}) = \sum_{t \in J}[1 - \mathsf{explains}_{\mathsf{full}}(\mathcal{M},t)] + 2 \cdot |\mathcal{M}| \quad (7)$$

$$= (m+1) \cdot \left(|U| - |\bigcup_{\theta_i \in \mathcal{M}} R_i| \right) + 2 \cdot |\mathcal{M}| \quad (8)$$

A mapping $\mathcal{M} \subseteq \mathcal{C}$ with $F(\mathcal{M}) \leq m = 2n$ thus exists if and only if $|\bigcup_{\theta_i \in \mathcal{M}} R_i| = |U|$ and $|\mathcal{M}| \leq n$, which is exactly the case where $\mathcal{M}$ encodes a covering selection with at most n sets. Furthermore, if such mappings exist, the optimal mapping according to (5) is one of them, and a polynomial time solution for mapping selection with full st tgds can thus be used to find a candidate solution that can be verified or rejected in polynomial time to answer SET COVER. □

4 Selection over ST TGDs

We now extend our approach to the complete language of st tgds with existentially quantified variables, showing how we assign credit for the shared null values such st tgds introduce. We begin by generalizing our two functions $\mathsf{error}_{\mathsf{full}}(\cdot)$ and $\mathsf{explains}_{\mathsf{full}}(\cdot)$ to model the partial evidence provided by st tgds with existentials. We then revisit our optimization problem using the new, more general functions.

4.1 Incomplete Errors

In contrast to $\mathsf{error}_{\mathsf{full}}(\cdot)$, an error function for arbitrary st tgds has to take into account incomplete tuples, that is, tuples with nulls created by a mapping with existentials.

Example 3: *The candidate θ_1 in Figure 1(d) is not valid with respect to the data example in Figure 1(b). However, if we add the tuple t_1 =task(BigData, Bob, 123) to J then θ_1 is valid for $(I, J \cup t_1)$. But this specific tuple is not in every $J' \supseteq J$ for which θ_1 is valid. Hence, t_1 is not a* full error. *However, a tuple k_1 =task(BigData, Bob, N_0) (where N_0 is a labeled null representing any constant) up to the renaming of the null must be in every such J'. Furthermore, such a tuple is in K_{θ_1}, the canonical universal solution for θ_1 over I.* □

Intuitively, for this example, a tuple in $K_{\mathcal{C}}$ should be an error if there is no homomorphism from that tuple to J. This is sufficient to consider k_1 to be an error for the original J of Figure 1(b), but not an error if we add t_1 to J. However, once an existentially quantified variable is shared between several atoms, we need a more general definition.

Example 4: *The candidate θ_3 in Figure 1(d) is not valid with respect to the extended data example in Figure 1(b)-(c). For it to be valid, J would have to contain two tuples k_1 =task(BigData, Bob, N_0) and k_2 =org(N_0, IBM) with a shared labeled null enabling the join on proj.lead. Suppose we add t_1 =task(BigData, Bob, 123) from above to J and t_2 =org(333, IBM) to J. If we just required each tuple in K_{θ_3} to have a homomorphism to some tuple in J, then neither would be considered an error, as there are homomorphisms from k_1 to t_1 and from k_2 to t_2. However, the instance $J \cup t_1 \cup t_2$ does not correctly connect Bob to IBM. Hence, we would like to consider both tuples to be errors.* □

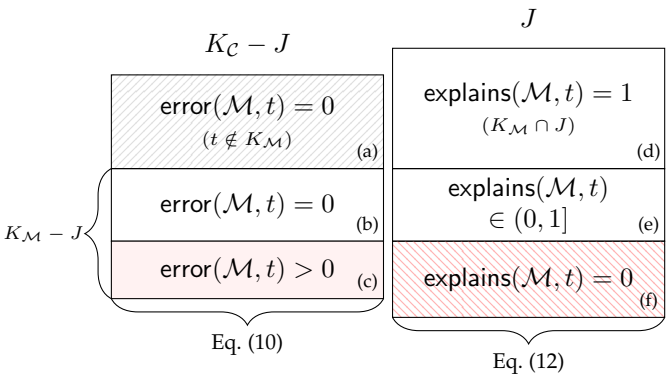


Fig. 3: Illustration of $\mathsf{explains}(\cdot)$ and $\mathsf{error}(\cdot)$ for selecting st tgds.

To address these issues, our $\mathsf{error}(\cdot)$ function is based on homomorphisms from all tuples in $K_{\mathcal{C}}$ resulting from a single chase step. If t is in the result of a chase step over $\theta = \forall x\, \phi(x) \rightarrow \exists y\, \psi(x,y)$, we call all (target) tuples resulting from this chase step (including t) the *context of t under θ* or *$context_\theta(t)$*.[2] We define the following helper function:

$$\mathsf{creates}(\theta,t) = \begin{cases} 0 & t \in K_{\mathcal{C}} - J,\ t \notin K_\theta \\ 0 & t \in K_\theta - J,\ \exists h : context_\theta(t) \rightarrow J \\ 1 & t \in K_\theta - J \text{ and no such } h \text{ exists} \end{cases} \quad (9)$$

Now for $\mathcal{M} \subseteq \mathcal{C}$ we define the $\mathsf{error}(\cdot)$ function as follows.

$$\mathsf{error}(\mathcal{M},t) = \sum_{\theta \in \mathcal{M}} \mathsf{creates}(\theta,t) \quad (10)$$

In Figure 3, which extends Figure 2 for selection over st tgds, $\mathsf{error}(\cdot)$ divides $K_{\mathcal{C}} - J$ into three parts for given $\mathcal{M}$: the tuples in (a) are created by no st tgd in $\mathcal{M}$, those in (b) do not count as errors because homomorphisms exist from them to J, and the remaining st tgds in (c) count as errors.

Recall that in the canonical universal instance $K_{\mathcal{C}}$ nulls are only shared between tuples generated by a single chase step. So each incomplete tuple $t \in K_{\mathcal{C}}$ (containing one or more nulls) is associated with a single chase step and st tgd θ. Hence, for such a tuple t, Equation (10) evaluates to 1 if there is no homomorphism from the $\mathsf{context}_\theta(t)$ to J, i.e., t is an error, and 0 otherwise. For a ground tuple t_g (with no nulls), if there is no homomorphism to J (meaning the tuple is not in J), Equation (10) counts how many candidates make this error.

4.2 Partially Explained Tuples

We now extend *explaining* to arbitrary st tgds. More precisely, we use tuples with labeled nulls coming from st tgds with existentially quantified variables to *partially explain* tuples in the target instance J through homomorphisms.

Example 5: *Consider θ_1 in Figure 1(d) and tuple t =task(BigData, Alice, 111) in Figure 1(b). θ_1 partially explains t via a homomorphism from k = task(BigData, Alice, N_1)*

2. For this to be well-defined, we require that each candidate st tgd θ is normalized into a set of smaller logically equivalent st tgds where only atoms that share existentials are retained in a single st tgd [1].

to t. In the absence of candidates that fully explain t, we might include θ_1 in our selection. □

To define partial explanation, we treat nulls that play a structural role in connecting information like constants. For a tuple $t \in J$ and a candidate θ, we call $k \in K_\theta$ a *possible explanation* for t under θ if there is a homomorphism $h : context_\theta(k) \to J$ with $h(k) = t$. Let $\mathsf{E}(t, \theta)$ be the set of all possible explanations for t under θ. We call a labeled null *unique* if it appears exactly once in $context_\theta(k)$. For $k \in \mathsf{E}(t, \theta)$, we define $\mathsf{null}(k)$ to be the number of unique nulls in k divided by the arity of k. So $\mathsf{null}(k) = 0$ if k contains only constants or labeled nulls used at least twice. Otherwise, $\mathsf{null}(k) > 0$. We say that k explains t to degree $1 - \mathsf{null}(k)$, and define the auxiliary function $\mathsf{covers}(\theta, t)$ for $t \in J$ based on the maximal degree to which t is explained by any tuple:

$$\mathsf{covers}(\theta, t) = \begin{cases} \max_{k \in \mathsf{E}(t,\theta)} (1 - \mathsf{null}(k)) & \mathsf{E}(t, \theta) \neq \emptyset \\ 0 & \mathsf{E}(t, \theta) = \emptyset \end{cases} \quad (11)$$

A mapping $\mathcal{M} \subseteq \mathcal{C}$ explains a tuple t as well as the best st tgd $\theta \in \mathcal{M}$ does.

$$\mathsf{explains}(\mathcal{M}, t) = \max_{\theta \in \mathcal{M}} \mathsf{covers}(\theta, t) \quad (12)$$

Equation (12) can be used to divide J into three parts (Figure 3) for a given $\mathcal{M}$: those tuples fully (d) or partially (e) explained through tuples in (b), and those that cannot be explained by $\mathcal{M}$ at all (f).

Using the same size function as for full st tgds, we define the general *mapping selection problem* as follows:
Given schemas **S**, **T**, a data example (I, J), and a set $\mathcal{C}$ of candidate st tgds

$$\textbf{Find} \quad \operatorname*{argmin}_{\mathcal{M} \subseteq \mathcal{C}} \sum_{t \in J} [(1 - \mathsf{explains}(\mathcal{M}, t))] + \sum_{t \in K_\mathcal{C} - J} [\mathsf{error}(\mathcal{M}, t)] + \mathsf{size}(\mathcal{M}) \quad (13)$$

The only difference with the case of full st tgds is that we now use notions of error and explaining suitable for st tgds with existentially quantified variables. In terms of Figure 3, we seek a small $\mathcal{M}$ that minimizes the error in (c) and maximizes the degree to which tuples in (d) and (e) are explained.

If all candidates are full, this optimization coincides with the one in (5), and so is NP-hard as well (see Section 3.4). In Section 5, we provide an efficient approximation algorithm for finding a high quality solution $\mathcal{M}$.

4.3 Example of Selection over ST TGDs

We extend the running example to illustrate objective (13). We use a reduced candidate set $\mathcal{C}' = \{\theta_1, \theta_3\}$ (Figure 1(d)) and the data in Figure 1(b)-(c), but omit the leader relation. A universal solution K_{θ_1} for I contains the task tuples (BigData, Bob, Null_1) and (ML, Alice, Null_2), while a solution K_{θ_3} contains the task tuples (BigData, Bob, Null_3) and (ML, Alice, Null_4) and the org tuples (Null_3, IBM) and (Null_4, SAP).

For θ_1, $\mathsf{creates}(\cdot)$ is 1 for tuple task(BigData, Bob, Null_1), and 0 for all other tuples, and $\mathsf{covers}(\cdot)$ is $2/3$ for task(ML, Alice, 111) and 0 otherwise. This is because task(ML, Alice, Null_2) partially explains the latter via a homomorphism mapping Null_2 to 111. Similarly, for θ_3, $\mathsf{creates}(\cdot)$ is 1 for task(BigData, Bob, Null_3) and org(Null_3,IBM), but 0 for task(ML, Alice, Null_4) and org(Null_4,SAP), which partially explain task(ML, Alice, 111) and org(111, SAP) to degree $3/3$ and $2/2$ respectively, via a homomorphism mapping Null_4 to 111, with corresponding values for $\mathsf{covers}(\cdot)$. The different subsets of candidate st tgds thus obtain the following values for the individual parts and the total of objective function (13).

$\mathcal{M}$	$\sum 1 - \text{explains}$	$\sum \text{error}$	size	(13)
$\{\}$	4	0	0	4
$\{\theta_1\}$	$3\,1/3$	1	3	$7\,1/3$
$\{\theta_3\}$	2	2	4	8
$\{\theta_1, \theta_3\}$	2	3	7	12

The minimal value for the objective is that of the empty mapping, that is, the complexity term fullfils its role of guarding against overfitting on too little data here. But we also see that $\{\theta_1\}$ is preferred over $\{\theta_3\}$, which in turn is preferred over $\{\theta_1, \theta_3\}$. The reason is that while θ_3 covers more tuples than θ_1, it also produces more errors and is larger. If we add data for at least five more projects X of the same kind as the ML one, i.e., pairs of tuples proj(X,N,1) and task(X,Alice,111), the preferred mapping is $\{\theta_3\}$, as the empty mapping cannot explain the new target tuples, θ_1 explains each to degree $2/3$, and θ_3 fully explains them (while no mapping introduces additional errors).

5 Probabilistic Mapping Selection

We now introduce Collective Mapping Discovery (**CMD**), our efficient solution for schema mapping selection, using techniques from the field of probabilistic modeling [27] and statistical relational learning (SRL) [28]. Specifically, **CMD** encodes the mapping selection objective (Equation (13)) as a program in probabilistic soft logic (PSL) [14], and uses PSL inference to instantiate and solve the optimization problem. Inference in PSL is highly scalable and efficient, as it avoids the combinatorial explosion inherent to relational domains (the relations $\mathsf{error}(\cdot)$ and $\mathsf{explains}(\cdot)$) by solving a convex optimization problem, while providing theoretical guarantees on solution quality with respect to the combinatorial optimum.

However, like the majority of SRL methods, PSL relies on a closed world assumption to ensure a well-defined probability distribution. While we will not entirely remove this restriction, we introduce *prioritized disjunctions*, a novel extension to PSL that allows for existentials over closed domains (the existence of an st tgd θ, in our case) while maintaining the convexity of inference, which makes it possible to encode and efficiently solve model selection problems such as the mapping selection problem.

5.1 Probabilistic Soft Logic

PSL [14] is a language for defining collective optimization problems in relational domains. It comes with an efficient and scalable solver for these problems. The key underlying idea is to (1) model desirable properties of the solution as first-order rules, (2) allow random variables to take on soft values between 0 and 1, rather than Boolean values 0 or 1,

and (3) let the system find a truth value assignment to all ground atoms in the domain that minimizes the sum of the *distance to satisfaction* of all ground instances of the rules.

A PSL program is a set of weighted rules:

$$w: \quad b_1(\boldsymbol{X}) \wedge \ldots \wedge b_n(\boldsymbol{X}) \rightarrow h_1(\boldsymbol{X}) \vee \ldots \vee h_m(\boldsymbol{X}) \quad (14)$$

where $\boldsymbol{X}$ is a set of universally-quantified variables, the $b_i(\boldsymbol{X})$ and $h_j(\boldsymbol{X})$ are atoms over (subsets of) the variables in $\boldsymbol{X}$, and w is a non-negative weight corresponding to the importance of satisfying the groundings of the rule. In first-order logic, a grounding of such a rule is satisfied if its body evaluates to false (0) or its head evaluates to true (1). PSL generalizes this into a rule's *distance to satisfaction*, which is defined as the difference of the truth values of the body and the head (set to zero if negative), and uses *soft truth values* from the interval $[0, 1]$ instead of Boolean ones. It relaxes the logical connectives using the *Lukasiewicz t-norm* and its *co-norm*, which is exact at the extremes, provides a consistent mapping for values in-between, and results in a convex optimization problem. Given an interpretation $\mathcal{I}$ of all ground atoms constructed from the predicates and constants in the program, the truth values of formulas are defined as follows.

$$\begin{aligned} \mathcal{I}(\ell_1 \wedge \ell_2) &= \max\{0, \mathcal{I}(\ell_1) + \mathcal{I}(\ell_2) - 1\} \\ \mathcal{I}(\ell_1 \vee \ell_2) &= \min\{\mathcal{I}(\ell_1) + \mathcal{I}(\ell_2), 1\} \\ \mathcal{I}(\neg l_1) &= 1 - \mathcal{I}(\ell_1) \end{aligned}$$

The distance to satisfaction of a ground rule $r = \text{body} \rightarrow \text{head}$ is defined as follows:

$$d_r(\mathcal{I}) = \max\{0, \mathcal{I}(\text{body}) - \mathcal{I}(\text{head})\} \quad (15)$$

Let R be the set of all ground rules obtained by grounding the program with respect to the given constants. The probability density function f over $\mathcal{I}$ is:

$$f(\mathcal{I}) = \frac{1}{Z} \exp[-\sum_{r \in R} w_r(d_r(\mathcal{I}))] \quad (16)$$

where w_r is the weight of rule r and Z is a normalization constant. PSL inference finds $\text{argmax}_{\mathcal{I}} f(\mathcal{I})$, that is, the interpretation $\mathcal{I}$ that minimizes the sum of the distances to satisfaction of all ground rules, each multiplied by the corresponding rule weight. Typically, truth values for some of the ground atoms are provided as evidence, that is, they have *observed* fixed truth values, and we only need to *infer* the optimal interpretation of the remaining atoms. PSL finds an exact optimum using soft truth values, which is then converted to a high quality discrete solution [14].

5.2 Mapping Selection in PSL

We now encode the mapping selection problem as a PSL program. We introduce three observed predicates that encode tuple membership in the target instance J and the $\mathsf{covers}(\cdot)$ and $\mathsf{creates}(\cdot)$ functions defined in Section 4, respectively, and one predicate in whose truth values denote membership of candidate st tgds in the selection, and thus need to be inferred by PSL. A given data example (I, J) and set of candidate st tgds $\mathcal{C}$ will introduce a constant for every tuple in $K_{\mathcal{C}} \cup J$ and for every candidate in $\mathcal{C}$. We use the logical variable F for st tgds, and T for tuples. The **CMD** program consists of the following rules:

$$\mathsf{size}(F): \quad in(F) \rightarrow \bot \quad (17)$$

$$1: \quad J(T) \rightarrow \exists F.\ \mathsf{covers}(F, T) \wedge in(F) \quad (18)$$

$$1: \quad in(F) \wedge \mathsf{creates}(F, T) \rightarrow J(T) \quad (19)$$

Rule (17) implements the size penalty by stating that we prefer not to include an st tgd in the selected set: its weighted distance to satisfaction is $\mathsf{size}(f) \cdot (\mathcal{I}(in(f)) - 0)$, and thus minimal if $in(f)$ is false. Rule (18) states that if a tuple is in J, there should be an st tgd in the set that covers that tuple, thus implementing the $\mathsf{explains}(\cdot)$ term. Note that the existential quantifier is not supported by PSL; we describe how we extend PSL and implement this efficiently in the next subsection. Rule (19) states that if an st tgd creates a tuple, that tuple should be in J, or conversely, if a tuple is not in J (and thus in $K_{\mathcal{C}} - J$), no st tgd in the selected set should create it. This implements the $\mathsf{error}(\cdot)$ penalty. The advantage of this approach is that it reasons about the interactions between tuples and st tgds in a fine-grained manner. In Section 5.4 we show the rules combine to implement the mapping selection objective (13).

5.3 Prioritized Disjunction Rules

In first-order logic (with finite domains), formulas with existential quantifiers, such as Rule (18) above, can be rewritten by expanding the existential quantifier into a disjunction over all groundings of its variables; however, in the context of PSL, the resulting disjunction of conjunctions in the head of a rule is expensive and non-convex to optimize in general. We therefore show how to efficiently handle a practically important subclass of such rules through a novel rewriting approach. We call these rules *prioritized disjunction rules*, as they implement a choice among groundings of an existentially quantified variable using observed soft truth values to express preferences or priorities over the alternatives (in the case of Rule (18), over st tgds to be selected). A prioritized disjunction rule is a rule:

$$w: \ b(X) \rightarrow \exists Y.\ h_o(Y, X) \wedge h_i(Y) \quad (20)$$

where $b(X)$ is a conjunction of atoms, $h_o(Y, X)$ is an observed atom and $h_i(Y)$ is an atom whose value will be inferred. In our case, see (18), $b(X)$ is $J(T)$, $h_o(Y, X)$ is $\mathsf{covers}(F, T)$, and $h_i(Y)$ is $in(F)$. The observed truth values of the $h_o(Y, X)$ atoms reflect how good a grounding of Y is for a grounding of X, as the truth value of the head will be higher when assigning high truth values to $h_i(Y)$ with high $h_o(Y, X)$. To efficiently support this comparison of alternatives, we introduce a *k-prioritization* for some natural number k, restricting the truth values of $h_o(Y, X)$ to $\{{}^0/_k, ..., {}^k/_k\}$ only. This allows us to rewrite each prioritized disjunction rule into a collection of rules, where we first expand the existential quantifier in the usual way, and then introduce a rule for each priority level.

Consider first the Boolean case, i.e., $k = 1$. In this case, every disjunct $h_o(Y, X) \wedge h_i(Y)$ is either false or equivalent to $h_i(Y)$. Since $h_o(Y, X)$ is observed, for every grounding y of Y, we can either drop the entire disjunct if $h_o(y, X)$ is false or drop $h_o(y, X)$ if it is true, leaving only $h_i(y)$ in the

disjunctive head. This leaves us with a standard PSL rule with a (possibly empty) disjunction of h_i atoms in the head.

For arbitrary k, we generalize this by grouping the head elements based on the priorities. For each grounding $b(x)$ of the rule body $b(X)$, we create one ground rule for every $j = 1, .., k$ of the following form:

$$w/k : b(x) \rightarrow \bigvee_{h_o(x,y) \geq j/k} h_i(y)$$

That is, we have a set of rules with identical bodies whose heads are progressively more general disjunctions of h_i atoms.

$$\begin{aligned} w/k : b(x) &\rightarrow \bigvee_{h_o(x,y) \in \{k/k\}} h_i(y) \\ w/k : b(x) &\rightarrow \bigvee_{h_o(x,y) \in \{k/k, (k-1)/k\}} h_i(y) \\ &\vdots \\ w/k : b(x) &\rightarrow \bigvee_{h_o(x,y) \in \{k/k, (k-1)/k, \ldots, 1/k\}} h_i(y) \end{aligned}$$

To understand the idea behind this transformation, assume for the moment that all $h_i(y)$ have fixed, Boolean truth values, and let m/k be the highest value $h_o(x, y)$ takes for this x and any y with $h_i(y) = 1$, i.e.,

$$m/k = \max_{\{y | h_i(y)=1\}} h_o(x, y)$$

Then, the rules for $j = 1, .., m$ are satisfied (because their head evaluates to 1), and the ones for $j = (m + 1), .., k$ are not satisfied (because their head evaluates to 0). More precisely, their distance to satisfaction is the truth value of $b(x)$, and each of these thus contributes $w/k \cdot \mathcal{I}(b(x))$ to the overall distance to satisfaction, which for this set of ground rules is

$$(k-m) \cdot w/k \cdot \mathcal{I}(b(x)) = w \cdot \left(1 - \max_{\{y | h_i(y)=1\}} h_o(x, y)\right) \cdot \mathcal{I}(b(x))$$

If b is observed, e.g., $\mathcal{I}(b(x)) = 1$ as in the case of (18), this expression depends purely on the maximum value of h_o.

Example 6: *Consider a single grounding of Rule* (18) *for* $t =$*org(111, SAP) in* J *from Figure 1(c) and the candidates* θ_3 *and* θ_4 *from Figure 1(d). The expanded ground rule is*

$$1 : \ \top \rightarrow \mathsf{covers}(\theta_3, t) \wedge in(\theta_3) \vee \mathsf{covers}(\theta_4, t) \wedge in(\theta_4)$$

Predicate org *has arity two, so we get a* 2*-prioritization with possible values* $\mathsf{covers}(F, t) \in \{0/2, 1/2, 2/2\}$. *Using values* $\mathsf{covers}(\theta_3, t) = 2/2$ *and* $\mathsf{covers}(\theta_4, t) = 1/2$, *we replace the initial ground rule with*

$$\begin{aligned} 1/2 : \ \top &\rightarrow in(\theta_3) \vee in(\theta_4) \\ 1/2 : \ \top &\rightarrow in(\theta_3) \end{aligned}$$

which completes the rewriting from a rule with existential quantification to a set of regular PSL rules. □

To summarize, we have shown an efficient transformation of a PSL rule with existentials over disjunctions of conjunctions in the head into a (compact) set of regular PSL rules using prioritized disjunctions. Furthermore, the soft-truth value semantics of the disjunction is the maximum over the disjuncts — which we will show to be a useful choice. While this extension was motivated by the mapping selection problem, we expect it to also be useful in other scenarios that involve choices between variable numbers of alternatives.

5.4 Objective Equivalence

Recall from Equation (13) that our goal is to minimize

$$\sum_{t \in J} [1 - \max_{\theta \in \mathcal{M}} \mathsf{covers}(\theta, t)] \tag{21}$$

$$+ \sum_{t \in K_{\mathcal{C}} - J} \left[\sum_{\theta \in \mathcal{M}} \mathsf{creates}(\theta, t) \right] \tag{22}$$

$$+ \sum_{\theta \in \mathcal{M}} \mathsf{size}(\theta) \tag{23}$$

We now demonstrate that, for Boolean values of the $in(\theta)$ atoms, this is exactly the objective used by our PSL program.

We get a grounding of Rule (17) for every st tgd $\theta \in \mathcal{C}$:

$$\mathsf{size}(\theta) : in(\theta) \rightarrow \bot \tag{24}$$

For $\theta \in \mathcal{M}$, this rule has distance to satisfaction 1, and 0 otherwise. Thus, each $\theta \in \mathcal{M}$ adds $\mathsf{size}(\theta)$ to PSL's distance to satisfaction, so those rules together correspond to (23). The error Rule (19) is trivially satisfied for tuples in J (and any st tgd). Thus, we only need to consider the groundings for $t \in K_{\mathcal{C}} - J$ and $\theta \in \mathcal{C}$:

$$1 : in(\theta) \wedge \mathsf{creates}(\theta, t) \rightarrow \bot \tag{25}$$

Such a ground rule has distance to satisfaction $\mathsf{creates}(\theta, t) - 0 = \mathsf{creates}(\theta, t)$. Recall from Equation (9) that this can only be non-zero for $t \in K_\theta - J$. The PSL sum thus adds $1 \cdot \mathsf{creates}(\theta, t)$ for every $\theta \in \mathcal{M}$ and $t \in K_\theta - J$, which equals (22). Rule (18) is trivially satisfied for $t \notin J$, and for every $t \in J$ results in a partially ground rule

$$1 : \top \rightarrow \exists F.\ \mathsf{covers}(F, t) \wedge in(F) \tag{26}$$

To complete the grounding, we apply PSL's prioritized disjunction rules. Recall (cf. Section 4.2) that the $\mathsf{covers}(\cdot)$ function takes on values according to the null function, which is the number of unique nulls divided by the arity of a tuple. Therefore, we know there are values $\{0/k, \ldots, k/k\}$ for $\mathsf{covers}(F, t)$ where k is the arity of the tuple t. Thus we get for each $t \in J$ a set of k ground rules, the ith of which is

$$1/k : \ \top \rightarrow \bigvee_{\theta \in \mathcal{C}, \mathsf{covers}(\theta, t) \geq i/k} in(\theta) \tag{27}$$

We know that for every $t \in J$, the associated groundings collectively contribute a distance to satisfaction of

$$1 - 1 \cdot \max_{\theta \in \mathcal{M}} \mathsf{covers}(\theta, t)$$

due to prioritized disjunction rules rewriting, which equals (21). Thus, the **CMD** program optimizes Equation (13).

5.5 Collective Mapping Discovery

To summarize, given data example (I, J) and candidates $\mathcal{C}$, **CMD** does the following two steps.

1) Compute truth values of evidence from (I, J) and $\mathcal{C}$
2) Perform PSL inference on the **CMD** program and evidence and return the corresponding mapping

Step 1 (*data preparation*) performs data exchange to determine the tuples in $K_{\mathcal{C}}$, and computes the truth values of the $|\mathcal{C}| \times |K_{\mathcal{C}} - J|$ many creates$(\cdot)$ atoms (based on Equation (9)) and of the $|\mathcal{C}| \times |J|$ many covers$(\cdot)$ atoms (based on Equation (11)). While finding a discrete solution to the optimization problem defined by the **CMD** program and evidence is NP-hard, Step 2 (***CMD** optimization*) provides an extremely scalable approximate solution with theoretical quality guarantees.

6 Evaluation

We experimentally evaluate **CMD** on a variety of scenarios, both synthetic and real world, showing that it robustly handles ambiguous and dirty metadata as well as dirty tuples in the data example, and scales well with the size of both metadata and data. We also demonstrate that our prioritized disjunction rules enable efficient inference for complex mapping scenarios. We ran our experiments on an Intel Xeon with 24 x 2.67GHz CPU and 128GB RAM. Our implementation of **CMD** and instructions for reproducing all experiments can be found online.[3]

6.1 Scenario Generation

Each of our scenarios consists of a data example (I, J) for a pair of schemas **S** and **T** and a set $\mathcal{C}$ of candidate st tgds, which form the input for **CMD**, and a gold standard mapping $\mathcal{M}_G$ used to assess the quality of the solution. Scenario generation uses the following steps:

1) We generate schemas **S** and **T**, correspondences, the initial data example (I, J_G), and a gold standard mapping $\mathcal{M}_G$ that is valid and fully explaining for (I, J_G) using the metadata generator iBench [18] and data generator ToxGene [29].
2) To create dirty metadata, we generate additional foreign key constraints and correspondences.
3) We use the implementation of the Clio [3] algorithm provided by ++Spicy [24] to generate the set of candidates $\mathcal{C}$ from the schemas, foreign key constraints and correspondences generated in previous steps, that is, based on metadata only.
4) We generate J starting from J_G, introducing errors and unexplained tuples with respect to $\mathcal{M}_G$.

The rest of this subsection provides more details on this process, and Table 1 lists the parameters controlling it. All experimental parameters are for scenario generation; we set their defaults and ranges to produce realistic scenarios.

Step 1. iBench uses transformation primitives to create realistic complex mapping scenarios. We chose a representative set of seven primitives[4]. One invocation of this set creates a total of eight source and ten target relations, and seven st tgds in $\mathcal{M}_G$. Three of those are full, the other four all contain existentials used once or twice and include existentials that are shared between relations. To create larger scenarios, we invoke the set $\pi_{Invocations}$ times. We set the size of I to $\pi_{TuplesPerTable}$ per relation.

Step 2. To obtain candidate st tgds with wrong join paths, we use iBench to add randomly created foreign keys to π_{FKPerc}% of the target relations. To obtain candidate st tgds making wrong connections between the schemas, we introduce additional correspondences as follows. We randomly select $\pi_{Corresp}$% of the target relations. For every selected target relation T, we randomly select a source relation S from those of the iBench primitive invocations not involving T (so Clio [23] can generate $\mathcal{M}_G$ as part of $\mathcal{C}$). For each attribute of T, we introduce a correspondence to a randomly selected attribute of S.

Step 4. We restrict data instance modifications to errors and explainable tuples with respect to $\mathcal{M}_G$, as unexplainable tuples can be removed prior to optimization (cf. Section 3.3). In our scenarios, $\mathcal{M}_G \subseteq \mathcal{C}$, and thus $K_G \subseteq K_{\mathcal{C}}$. So each tuple in $K_{\mathcal{C}}$ is either generated by both $\mathcal{M}_G$ and $\mathcal{C}-\mathcal{M}_G$, only by $\mathcal{M}_G$ (i.e., an error tuple if deleted from J), or only by $\mathcal{C}-\mathcal{M}_G$ (i.e., a new explainable tuple if added to J). As tuples in $K_{\mathcal{C}}$ may have nulls, we take into account homomorphisms when determining which of these cases applies to a given tuple. We randomly select a set J_{New} containing $\pi_{Explainable}$% of the potential new explainable tuples, and a set J_{Err} containing π_{Errors}% of the potential error tuples, and set $J = J_G \cup J_{New} \setminus J_{Err}$.

TABLE 1: Overview of main experimental parameters.

Parameter	Range	Default
$\pi_{Invocations}$	1 - 10	2
$\pi_{TuplesPerTable}$	10 - 100	50
π_{FKPerc}	0 - 10%	0%
$\pi_{Corresp}$	0 - 100%	0%
π_{Errors}	0 - 30%	0%
$\pi_{Explainable}$	0 - 100%	0%

6.2 Evaluation of Solution Quality

We assess quality by comparing the mapping $\mathcal{M} \subseteq \mathcal{C}$ selected by **CMD** to (1) the correct mapping $\mathcal{M}_G$ produced by iBench and (2) the set $\mathcal{C}$ of all candidates produced by Clio, which serves as our metadata-only baseline. Since Clio does not use data, we are not confounding in our experiments how **CMD** uses data with other proposed approaches. Directly comparing mappings is a hard problem, so we follow the standard in the literature which is to compare the data exchange solutions produced by the mappings [30].

3. http://projects.linqs.org/project/cmd

4. CP copies a source relation to the target, changing its name. ADD copies a source relation and adds attributes; DL does the same, but removes attributes instead; and ADL adds and removes attributes to the same relation. The number that are added or removed are controlled by range parameters, which we set to (2,4). ME copies two relations, after joining them, to form a target relation. VP copies a source relation to form two, joined, target relations. VNM is the same as VP but introduces an additional target relation to form a N-to-M relationship between the other target relations.

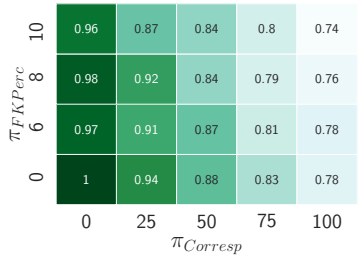


Fig. 4: Mapping quality (IQ-Score) for the Clio baseline with ambiguous metadata. **CMD** always reaches IQ-Score 1.

Fig. 5: Mapping quality (IQ-Score) for **CMD** with dirty data.

We use the core data exchange algorithm of ++Spicy [24] to obtain $K_{\mathcal{M}}$ and $K_{\mathcal{C}}$. The gold standard instance K_G for $\mathcal{M}_G$ is the original target instance J obtained from iBench in the first step. We compare these instances using the IQ-METER [19] quality measure. IQ-METER measures the ability of a mapping to generate correct tuples as well as correct relational structures via labeled nulls or invented values, so it is appropriate as an evaluation measure for our mappings which contain existentials. It calculates recall and precision of tuples and recall and precision of joins. The distance between mappings is then defined as one minus the harmonic mean of these four measures; for full details, see Mecca et al [31]. We directly use the harmonic mean, which we call IQ-score$(K_1, K_2) \in [0, 1]$, where higher is better.

6.3 CMD Accuracy over Ambiguous Metadata

We begin by assessing the ability of **CMD** to handle ambiguous or dirty metadata and still identify a good mapping from the set of candidates. We increase the number of candidate st tgds by increasing the π_{FKPerc} parameter from 0 to 10 percent and the $\pi_{Corresp}$ parameter from 0 to 100 percent. We use five scenarios per parameter setting, with an average of 800 source and 1000 target tuples. **CMD** always found the correct mapping, i.e., it resolved all metadata ambiguities based on the data example. In contrast, for Clio, which uses metadata only, the IQ-scores decreased with more imprecise evidence, as shown in Figure 4.

6.4 CMD Accuracy over Dirty Data

Our second experiment investigates the effect of imperfect data on mapping quality. We vary the percentage π_{Errors} of added errors from 0 to 30% in steps of five, and the percentage $\pi_{Explainable}$ of additional explainable tuples from 0 to 100% in steps of 25. We set $\pi_{Corresp} = 100\%$ to maximize the number of potential explainable but undesired tuples. We consider five scenarios in each case, with 800 source tuples and 1000 tuples in the initial target instance J. The numbers of additional tuples obtained range from zero to 300 for errors and from zero to 1800 for explainable tuples.

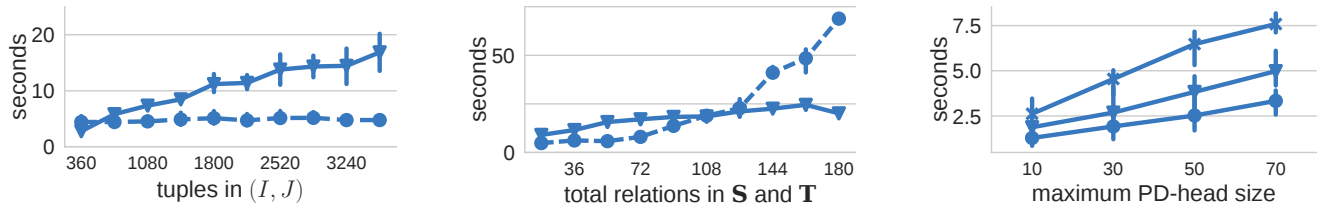
In Figure 5, we plot IQ-score as we vary π_{Errors} and $\pi_{Explainable}$. Generally, as the number of errors increases, i.e., more correct tuples are missing from the target instance, the quality of the mapping selected by **CMD** decreases, as there is less incentive to include candidates that would correctly explain such tuples, which results in lower IQ-score due to lower recall. Adding explainable tuples also generally decreases the quality of the mapping, as they provide incentives to include additional st tgds that, while explaining those dirty tuples, generally decrease precision and thus IQ-score. However, in the presence of significant numbers of errors, explainable tuples increase mapping quality. This happens whenever explainable tuples cause **CMD** to select st tgds in $\mathcal{C}-\mathcal{M}_G$ that are similar to $\mathcal{M}_G$, e.g., omitting a target join on an existential variable, and when selecting those st tgds is preferred over the empty mapping. For scenarios with over one quarter additional explainable tuples, and even in the presence of a few (less than 10%) errors, **CMD** routinely finds mappings with high IQ-scores. This confirms that the fine-grained optimization score handles increasing noise levels gracefully.

6.5 Performance of CMD

The next set of experiments evaluates the performance of our approach along several dimensions. We focus on *optimization time*, i.e., the time to find an optimal mapping after data preparation is completed. Data preparation (determining which tuples are errors or unexplained for each st tgd in $\mathcal{C}$) is slow, taking up to 150 minutes in our largest scenarios; it would be easy to optimize this time, however that is not a focus of our current work.

DATA SIZE. We vary $\pi_{TuplesPerTable}$ from 10 to 100 in steps of 10 tuples to obtain data examples of increasing size for our default schema size of 36 relations. We generate five scenarios for each setting. Figure 6(a) plots the average optimization time in **CMD** and average time to generate $\mathcal{C}$ (the Clio baseline). **CMD** optimization times are comparable to Clio times even though we optimize over relatively large (3600 tuple) data examples.

SCHEMA SIZE. We vary $\pi_{Invocations}$ from 1 to 10 to increase the sizes of the schemas (and thus the number of candidate st tgds that are plausible for the schemas), which results in source and target schemas with 18 to 180 relations total. The largest scenario involved 150 candidate st tgds, or over 10^{45} possible mappings. We set $\pi_{TuplesPerTable} = 50$, thus obtaining data examples with 900 to 9000 tuples. We use five scenarios per setting and, as before, plot average **CMD** optimization time and average time to run Clio (Figure 6(b)). Again, **CMD** optimization times are comparable to those of Clio, but for increasing schema size, the latter increases more quickly.



(a) (b) (c)

Fig. 6: Optimization time of **CMD** w.r.t. (a) size of data example, (b) schema sizes, and (c) maximal number of candidates explaining the same tuple. In (a) and (b), the dashed line is for the Clio baseline. In (c), $\pi_{SchemaSize}$ is 20 (top), 10 (middle) or 5 (bottom).

COMPLEXITY. We vary the maximal number π_{PDSize} of candidate st tgds that explain each target tuple, which corresponds to the number of head atoms in prioritized disjunction rules. This is the main parameter determining the complexity of optimization. Our custom-made scenarios use a single primitive, and their complexity is controlled through two parameters, the arity π_{Arity} and the schema size $\pi_{SchemaSize}$. The source has $\pi_{SchemaSize}$ relations of arity π_{Arity}, the target $\pi_{SchemaSize} \cdot \pi_{Arity}$ relations of arity at most π_{Arity}. The gold standard mapping has $\pi_{SchemaSize}$ st tgds, and the number of potential candidates increases quadratically with both $\pi_{SchemaSize}$ and π_{Arity}. We set $\pi_{TuplesPerTable} = 25$.

We consider all combinations of $\pi_{SchemaSize} \in \{5, 10, 20\}$ and $\pi_{Arity} \in \{5, 10, 20\}$, and vary π_{PDSize} from 10 to 70 in steps of 20. We use one scenario for each combination. In Figure 6(c), we plot the optimization time for each value of $\pi_{SchemaSize}$, aggregating over π_{Arity}. In all scenarios, the mapping selected by **CMD** has perfect IQ-score. This result shows that optimization with prioritized disjunction rules is efficient even with 70 candidates explaining the same tuples, an order of magnitude higher than seen in our other tests.

6.6 CMD on Real Metadata and Data

The previous results show the power of **CMD** on benchmark datasets. Next we consider several real world scenarios.

AMALGAM. We first consider the well-known Amalgam benchmark [32], using schema *A1* as source and *A3* as target. To construct a data example, we select a small subset of the data in A1. We use ideal correspondences, so this problem tests whether **CMD** selects st tgds with correct joins from the candidates generated by Clio. The final evaluation contains 18 relations and 2,502 tuples. For this scenario, **CMD** achieves IQ-score .99 and optimization time was under a minute. **NEUROSCIENCE.** We map Allen Brain Atlas (ABA), Kyoto Encyclopedia of Genes and Genomes (KEGG), Pharmacogenomics Knowledgebase (PharmGKB), and UniProt (Universal Protein resource) schemas to the Semantic MediaWiki Linked Data Environment (SMW-LDE) Ontology [33], [34]. ABA has one relation and 15 tuples; KEGG has four relations and 56 tuples; PharmGKB has four relations and 142 tuples; and UniProt has one relation and 15 tuples. The common target schema has 31 relations and 54 foreign keys. As with Amalgam, we construct data examples from the source instances and we use ideal correspondences. The **CMD** mappings for ABA, PharmGKB and UniProt achieved perfect IQ-scores. For KEGG, **CMD** got a score of .93.

CMD achieved a lower score on KEGG because it selected some candidates that reused labeled nulls for some attributes where the gold standard exchanged variables from the source. Our current scoring function cannot distinguish these, but could easily be adapted to do so.

7 RELATED WORK

USING METADATA. Using metadata information to guide schema mapping discovery has a long tradition. The names of schema elements (such as attributes) can be used to suggest attribute correspondences (the well-known schema matching problem) and the Clio project showed how the schemas and constraints can be used to infer mappings [2], [3]. HepTox [26] and ++Spicy [24] have extended this to richer forms of metadata (including equality-generating-dependencies). In addition, the role of data in resolving ambiguity or incompleteness in the metadata evidence has long been recognized, both in matching [35] and in schema mapping, where the Data Viewer [21] and Muse [5] systems use source and target data interactively to help a user understand, refine or correct automatically generated mappings. Most systems that combine evidence from metadata and data, do so heuristically and may fail to suggest a good mapping under inconsistent evidence.

USING DATA EXAMPLES. A complementary approach that uses data only is often called example-driven schema-mapping design [36]. An example that is closest to ours casts schema mapping discovery from data as a formal (and fully automated) learning problem [7]. Given a single data example (I, J) find a mapping M that is valid and fully explaining (and of minimal size) for (I, J). Even for full st tgds finding optimal mappings in this framework is DP-hard [7]. Using this framework, ten Cate et al. [11] consider the restricted class of mappings with a single atom (relation) in the head and in the body. They provide a greedy approximation algorithm that is guaranteed to find near optimal (valid and fully explaining) mappings of this type, but do not discuss experimental results.

In contrast, we do not require the mapping to be valid or fully explaining, rather we define an optimization problem that finds an optimal set of st tgds that minimizes errors (the

invalidity of a mapping) and unexplained tuples. Although the number of errors can, in theory, be exponential in the size of the mappings (as pointed out by Gottlob and Senellart [7]), we manage this complexity by using a set of candidate st tgds derived from real mapping discovery systems and by using an efficient approximation framework (PSL) to reason over these alternative mappings. We also provide a novel principled way of combining evidence from mappings that contain existentials (and hence only partially explain tuples).

MULTIPLE EXAMPLES. The Eirene system returns the most general mapping that fits a set of data examples, if one exists, and otherwise guides a user in refining her data (not the mapping) to identify tuples that are causing the failure [22].

This is in contrast to Muse and the Data Viewer systems that interactively pick data to help a user refine a mapping. Alexe et al. [12] have also studied when a mapping can be uniquely characterized by a set of data examples. This problem has also been cast as a learning problem, where a user labels a series of examples as positive or negative [37]. Finally, Sarma et al. [38] consider how to learn views (or full GAV mappings) from data alone.

RELATED SELECTION PROBLEMS. Belhajjame et al. [13] use feedback from users on exchange solutions to estimate the precision and recall of views. They present view selection as an optimization problem that maximizes either precision (which is maximal for valid mappings) or recall (which is maximal for fully explaining mappings), without taking mapping size into account. While Belhajjame et al. [13] do not provide runtimes for their approach, they use a very powerful, general purpose search algorithm [39] designed for constrained optimization problems. In contrast, our mapping selection problem, though NP-hard, is of a form for which PSL efficiently finds a high quality solution. Other work finds the top-k best matchings [40], while CMD finds only the best mapping. It is an open problem how to extend our optimization to top-k mappings.

While our approach relies on a potentially noisy data example (I, J) to select among mappings, Belhajjame et al. [13] rely on potentially noisy feedback from a user, who annotates target tuples in a query answer as expected (with respect to an implicit J) or unexpected, or provides additional expected tuples. User feedback has also been used in active learning scenarios in the context of data integration, e.g., to select consistent sets of attribute-attribute matches among many datasources [41], or to select join associations in the context of keyword-search based data integration [42]. While those settings are quite different from the one we consider here, extending **CMD** with active learning to incorporate additional feedback is an interesting direction for future work.

Similarly, the source selection problem [43] has been modeled as a problem of finding a set of sources that minimize the cost of data integration while maximizing the gain (a score that is similar to recall). Dong et al. [43] use the greedy randomized adaptive search procedure metaheuristic to solve the source selection problem, a heuristic which unlike PSL does not provide any approximation guarantees on the solution.

PROBABILISTIC REASONING. Statistical relational techniques have been applied to a variety of data and knowledge integration problems. Perhaps closest to our approach is the use of Markov Logic [44] for ontology alignment [45] and ontological mapping construction [46]. However, we consider more expressive mappings than either of those approaches. Furthermore, by using PSL, we can easily integrate partial evidence from st tgds with existential quantification through soft truth values. More importantly, in contrast to Markov Logic, PSL avoids the hard combinatorial optimization problem and instead provides scalable inference with guarantees on solution quality. This advantage has proven crucial also for applications of PSL in knowledge graph identification [15] and data fusion [16], [17].

8 CONCLUSION

We introduce *Collective Mapping Discovery* (**CMD**), a new approach to schema mapping selection that finds a set of st tgds that best explains the data in the sources being integrated. We use both metadata and data as evidence to resolve ambiguities and incompleteness in the sources, allowing some inconsistencies and choosing a small set of mappings that work collectively to explain the data. To solve this problem, we use and extend probabilistic soft logic (PSL), casting the problem as efficient joint probabilistic inference. The declarative nature of the PSL program makes it easy to extend **CMD** to include additional forms of evidence and constraints, coming from the domain, from user feedback, or other sources. In future work, we plan to explore weight learning techniques and investigate their impact in different problem settings.

Acknowledgments: AK has been supported by the Research Foundation Flanders (FWO). RJM is supported by NSERC, and LG by the National Science Foundation [IIS1218488]. The authors thank Boris Glavic, Patricia Arocena, Gianni Mecca, Donatello Santoro, and Radu Ciucanu for valuable help with iBench and ++Spicy; and Craig Knoblock for the Neuroscience problem set.

REFERENCES

[1] B. ten Cate and P. G. Kolaitis, "Structural characterizations of schema-mapping languages," *Commun. ACM*, vol. 53, no. 1, pp. 101–110, 2010.
[2] R. J. Miller, L. M. Haas, and M. Hernández, "Schema Mapping as Query Discovery," in *VLDB*, 2000.
[3] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, and R. Fagin, "Translating Web Data," in *VLDB*, 2002.
[4] H. Elmeleegy, A. K. Elmagarmid, and J. Lee, "Leveraging query logs for schema mapping generation in U-MAP," in *SIGMOD*, 2011.
[5] B. Alexe, L. Chiticariu, R. J. Miller, and W.-C. Tan, "Muse: Mapping understanding and design by example," in *ICDE*, 2008.
[6] B. ten Cate, P. G. Kolaitis, and W.-C. Tan, "Schema mappings and data examples," in *EDBT*, 2013.
[7] G. Gottlob and P. Senellart, "Schema mapping discovery from data instances," *Journal of the ACM (JACM)*, vol. 57, no. 2, p. 6, 2010.
[8] L. Qian, M. J. Cafarella, and H. Jagadish, "Sample-driven schema mapping," in *SIGMOD*, 2012.
[9] A. Bonifati, G. Mecca, A. Pappalardo, S. Raunich, and G. Summa, "Schema mapping verification: the spicy way," in *EDBT*, 2008.
[10] B. Alexe, B. ten Cate, P. G. Kolaitis, and W.-C. Tan, "Designing and refining schema mappings via data examples," in *SIGMOD*, 2011.
[11] B. ten Cate, P. G. Kolaitis, K. Qian, and W.-C. Tan, "Approximation algorithms for schema-mapping discovery from data examples," in *AMW*, 2015.

[12] B. Alexe, B. ten Cate, P. G. Kolaitis, and W.-C. Tan, "Characterizing schema mappings via data examples," *ACM Trans. Database Syst.*, vol. 36, no. 4, p. 23, 2011.
[13] K. Belhajjame, N. W. Paton, S. Embury, A. A. Fernandes, and C. Hedeler, "Incrementally improving dataspaces based on user feedback," *Information Systems*, 2013.
[14] S. H. Bach, M. Broecheler, B. Huang, and L. Getoor, "Hinge-loss markov random fields and probabilistic soft logic," *ArXiv:1505.04406 [cs.LG]*, 2015.
[15] J. Pujara, H. Miao, L. Getoor, and W. Cohen, "Knowledge graph identification," in *ISWC*, 2013.
[16] S. Fakhraei, B. Huang, L. Raschid, and L. Getoor, "Network-based drug-target interaction prediction with probabilistic soft logic," *IEEE/ACM Trans. Comput. Biology Bioinform.*, vol. 11, no. 5, pp. 775–787, 2014.
[17] P. Kouki, S. Fakhraei, J. Foulds, M. Eirinaki, and L. Getoor, "HyPER: A flexible and extensible probabilistic framework for hybrid recommender systems," in *RecSys*, 2015.
[18] P. C. Arocena, B. Glavic, R. Ciucanu, and R. J. Miller, "The iBench Integration Metadata Generator," *PVLDB*, vol. 9, no. 3, pp. 108–119, 2015.
[19] G. Mecca, P. Papotti, and D. Santoro, "IQ-METER - an evaluation tool for data-transformation systems," in *ICDE*, 2014.
[20] A. Kimmig, A. Memory, R. J. Miller, and L. Getoor, "A collective, probabilistic approach to schema mapping," in *ICDE*, 2017.
[21] L.-L. Yan, R. J. Miller, L. Haas, and R. Fagin, "Data-Driven Understanding and Refinement of Schema Mappings," in *SIGMOD*, 2001.
[22] B. Alexe, B. ten Cate, P. G. Kolaitis, and W.-C. Tan, "EIRENE: Interactive design and refinement of schema mappings via data examples," *PVLDB*, vol. 4, no. 12, pp. 1414–1417, 2011.
[23] R. Fagin, L. M. Haas, M. A. Hernández, R. J. Miller, L. Popa, and Y. Velegrakis, "Clio: Schema Mapping Creation and Data Exchange," in *Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos*, 2009, pp. 198–236.
[24] B. Marnette, G. Mecca, P. Papotti, S. Raunich, and D. Santoro, "++Spicy: an OpenSource Tool for Second-Generation Schema Mapping and Data Exchange," *PVLDB*, vol. 4, no. 12, pp. 1438–1441, 2011.
[25] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa, "Data exchange: semantics and query answering," *Theor. Comput. Sci.*, vol. 336, no. 1, pp. 89–124, 2005.
[26] A. Bonifati, E. Q. Chang, T. Ho, V. S. Lakshmanan, and R. Pottinger, "HePToX: Marrying XML and Heterogeneity in Your P2P Databases," in *VLDB*, 2005.
[27] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.
[28] L. Getoor and B. Taskar, Eds., *An Introduction to Statistical Relational Learning*. MIT Press, 2007.
[29] D. Barbosa, A. O. Mendelzon, J. Keenleyside, and K. A. Lyons, "Toxgene: a template-based data generator for XML," in *SIGMOD*, 2002.
[30] Z. Bellahsene, A. Bonifati, F. Duchateau, and Y. Velegrakis, "On evaluating schema matching and mapping," in *Schema matching and mapping*. Springer, 2011, pp. 253–291.
[31] G. Mecca, P. Papotti, S. Raunich, and D. Santoro, "What is the IQ of your data transformation system?" in *ACM CIKM*, 2012, pp. 872–881.
[32] R. J. Miller, D. Fisla, M. Huang, D. Kymlicka, F. Ku, and V. Lee, "The Amalgam Schema and Data Integration Test Suite," 2001, www.cs.toronto.edu/~miller/amalgam.
[33] C. Becker, C. Bizer, M. Erdmann, and M. Greaves, "Extending SMW+ with a linked data integration framework," in *ISWC*, 2011.
[34] C. Knoblock, P. Szekely, J. Ambite, A. Goel, S. Gupta, K. Lerman, M. Muslea, M. Taheriyan, and P. Mallick, "Semi-automatically mapping structured sources into the semantic web," in *ESWC*, 2012.
[35] J. Kang and J. F. Naughton, "On schema matching with opaque column names and data values," in *SIGMOD*, 2003.
[36] B. ten Cate, P. G. Kolaitis, and W. C. Tan, "Schema mappings and data examples," in *EDBT*, 2013.
[37] B. ten Cate, V. Dalmau, and P. G. Kolaitis, "Learning schema mappings," *ACM Trans. Database Syst.*, vol. 38, no. 4, p. 28, 2013.
[38] A. D. Sarma, A. G. Parameswaran, H. Garcia-Molina, and J. Widom, "Synthesizing view definitions from data," in *ICDT*, 2010.
[39] C. Audet and J. E. Dennis Jr., "Mesh adaptive direct search algorithms for constrained optimization," *SIAM Journal on Optimization*, vol. 17, no. 1, pp. 188–217, 2006.
[40] A. Gal, "Managing uncertainty in schema matching with top-k schema mappings," *Journal on Data Semantics VI: Special Issue on Emergent Semantics*, pp. 90–114, 2006.
[41] N. Q. V. Hung, N. T. Tam, Z. Miklós, K. Aberer, A. Gal, and M. Weidlich, "Pay-as-you-go reconciliation in schema matching networks," in *ICDE*, 2014.
[42] Z. Yan, N. Zheng, Z. G. Ives, P. P. Talukdar, and C. Yu, "Actively soliciting feedback for query answers in keyword search-based data integration," *PVLDB*, vol. 6, no. 3, pp. 205–216, 2013.
[43] X. L. Dong, B. Saha, and D. Srivastava, "Less is more: Selecting sources wisely for integration," *PVLDB*, vol. 6, no. 2, pp. 37–48, 2012.
[44] M. Richardson and P. Domingos, "Markov logic networks," *Machine Learning*, vol. 62, no. 1-2, pp. 107–136, 2006.
[45] M. Niepert, C. Meilicke, and H. Stuckenschmidt, "A probabilistic-logical framework for ontology matching," in *AAAI*, 2010.
[46] C. Zhang, R. Hoffmann, and D. S. Weld, "Ontological smoothing for relation extraction with minimal supervision," in *AAAI*, 2012.

Angelika Kimmig is a lecturer in the School of Computer Science and Informatics at Cardiff University. Her research interests include symbolic AI, reasoning under uncertainty, machine learning, logic programming, and especially combinations thereof such as probabilistic programming and statistical relational learning. She received her PhD in Computer Science from KU Leuven and her Diplom in Computer Science from the Albert-Ludwigs-Universität Freiburg.

Alex Memory is a student in the Department of Computer Science at University of Maryland and a research scientist at Leidos. His research interests include machine learning, data integration and anomaly detection. He received his BS in Computer Engineering from North Carolina State University and MS in Applied Math from Johns Hopkins University.

Renée J. Miller is a University Distinguished Professor of Computer Science at Northeastern University. She is a Fellow of the Royal Society of Canada, Canada's National Academy of Scienc. She received the US Presidential Early Career Award for Scientists and Engineers (PECASE), the highest honor bestowed by the United States government on outstanding scientists and engineers beginning their careers. She received an NSF CAREER Award and the Ontario Premier's Research Excellence Award. She formerly held the Bell Canada Chair of Information Systems at the University of Toronto and is a fellow of the ACM. She and her co-authors (Fagin, Kolaitis and Popa) received the (10 Year) ICDT Test-of-Time Award for their influential 2003 paper establishing the foundations of data exchange. She is Editor-in-Chief of VLDB Journal and was formerly president of the VLDB Foundation. She received her PhD in Computer Science from the University of Wisconsin, Madison and bachelor's degrees in Mathematics and Cognitive Science from MIT.

Lise Getoor is a professor in the Computer Science Department at the University of California, Santa Cruz. Her research areas include machine learning, data integration and reasoning under uncertainty, with an emphasis on graph and network data. She is a Fellow of the Association for Artificial Intelligence, an elected board member of the International Machine Learning Society, serves on the board of the Computing Research Association (CRA), and was co-chair for ICML 2011. She is a recipient of an NSF Career Award and eleven best paper and best student paper awards. She received her PhD from Stanford University in 2001, her MS from UC Berkeley, and her BS from UC Santa Barbara, and was a professor in the Computer Science Department at the University of Maryland, College Park from 2001-2013.